# Computer Graphics

CS 217

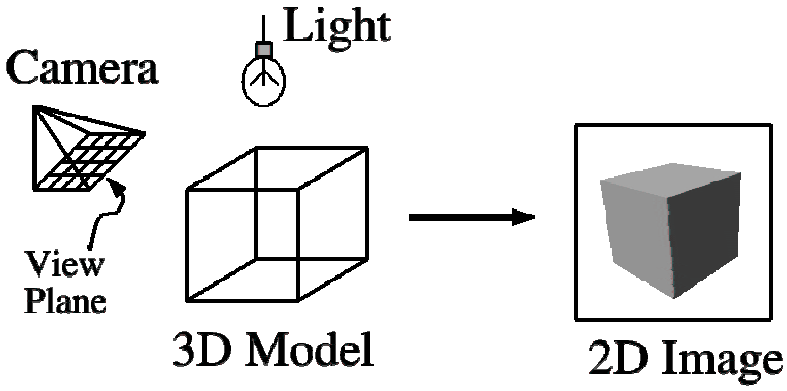
## Overview

- Introduction
  - What is computer graphics?
- Applications
  - What is it good for?
- Systems & software
  - How does it related to this course?

## Introduction



- What is computer graphics?



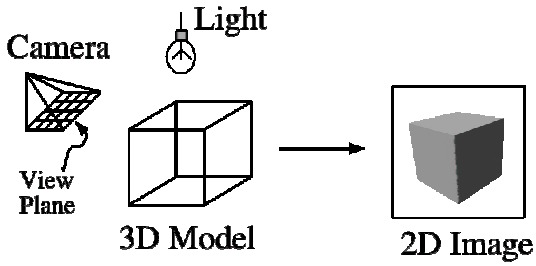

## Luxo, Jr.




What issues must be addressed by a computer graphics system?

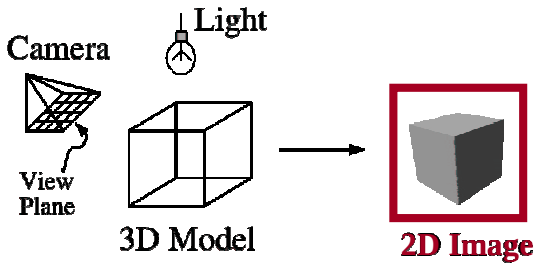
## Overview

- Topics in computer graphics
  - Imaging = *representing 2D images*
  - Modeling = *representing 3D objects*
  - Rendering = *constructing 2D images from 3D models*
  - Animation = *simulating changes over time*




## Overview

- Topics in computer graphics
  - Imaging = *representing 2D images*
  - Modeling = *representing 3D objects*
  - Rendering = *constructing 2D images from 3D models*
  - Animation = *simulating changes over time*



## What is an Image?

- An image is a 2D rectilinear array of pixels

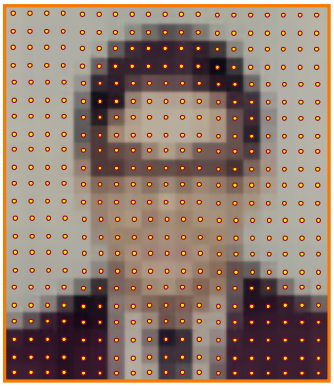
Continuous image

Digital image

## What is an Image?

- An image is a 2D rectilinear array of pixels

Continuous image



Digital image

A pixel is a sample, not a little square!

## What is an Image?

- An image is a 2D rectilinear array of pixels

Continuous image

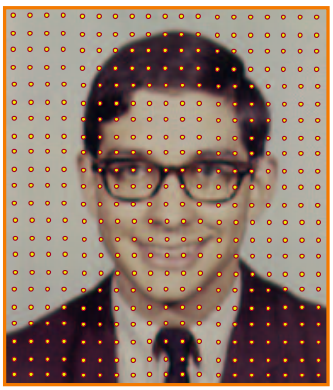
Digital image

A pixel is a sample, not a little square!

## Representing Digital Images

```
typedef struct pixel {
  float red, green, blue;
} Pixel;

typedef struct image {
  int width, height;
  Pixel *pixels;
} Image;
```



Digital image

# Image Abstract Data Type

```
/* Create and delete image */
Image *Image_new(int width, int height);
void Image_free(Image *image);

/* Get image properties */
int Image_getWidth(Image *image);
int Image_getHeight(Image *image);

/* Pixel manipulation */
Pixel *Image_getPixel(Image *image, int i, int j);
void Image_setPixel(Image *image, int i, int j, Pixel *pixel);

/* Image manipulation */
void Image_brighten(Image *image, float factor);
void Image_blur(Image *image, float factor);
void Image_swirl(Image *image, float angle);
void Image_composite(Image *image, Image *image2, Image *mask);
void Image_morph(Image *image1, Image *image2, Array *features);

etc.
```

# Processing Digital Images

- Examples
  - Filtering
  - Warping
  - Composition
  - Morphing

Image Warping

Image Composition

*(Michael Bostock, CS426, Fall99)*

Image Morphing

*(All students in CS 426, Fall98)*

## Processing Digital Images



- Example from the movies …

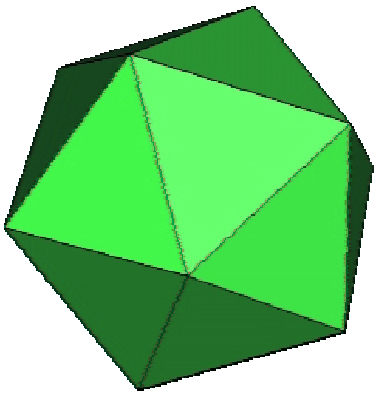
The Matrix Reloaded
*(Warner)*

## Overview



- Topics in computer graphics
  - Imaging = *representing 2D images*
  - Modeling = *representing 3D objects*
  - Rendering = *constructing 2D images from 3D models*
  - Animation = *simulating changes over time*

## Geometric Modeling



How can this object be represented in a computer?

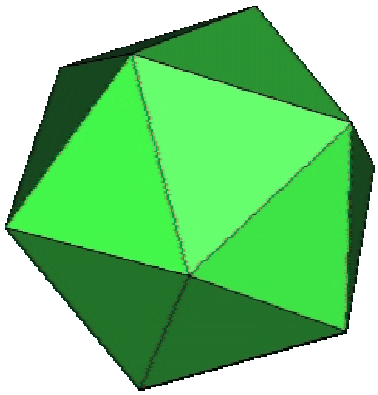
## Example: 3D Polygons

```
typedef struct point {
  float x, y, z;
} Point;

typedef struct polygon {
  int npoints;
  Point *points;
} Polygon;

typedef struct object {
  int npolygons;
  Polygon *polygons;
} Object;

typedef struct scene {
  int nobjects;
  Object *objects;
} Scene;
```



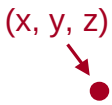
## Example: 3D Polygons

```
typedef struct point {
  float x, y, z;
} Point;

typedef struct polygon {
  int npoints;
  Point *points;
} Polygon;

typedef struct object {
  int npolygons;
  Polygon *polygons;
} Object;

typedef struct scene {
  int nobjects;
  Object *objects;
} Scene;
```



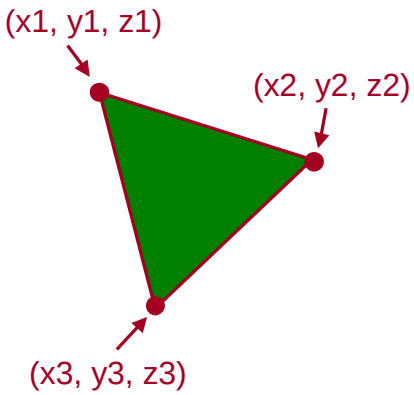

## Example: 3D Polygons

```
typedef struct point {
  float x, y, z;
} Point;

typedef struct polygon {
  int npoints;
  Point *points;
} Polygon;

typedef struct object {
  int npolygons;
  Polygon *polygons;
} Object;

typedef struct scene {
  int nobjects;
  Object *objects;
} Scene;
```



## Example: 3D Polygons

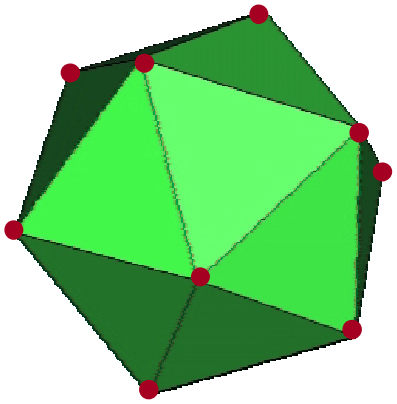


```
typedef struct point {
  float x, y, z;
} Point;

typedef struct polygon {
  int npoints;
  Point *points;
} Polygon;

typedef struct object {
  int npolygons;
  Polygon *polygons;
} Object;

typedef struct scene {
  int nobjects;
  Object *objects;
} Scene;
```



## Example: 3D Polygons

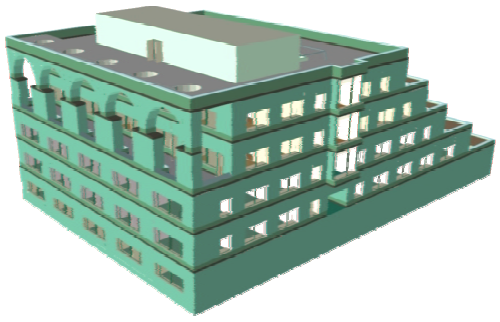


```
typedef struct point {
  float x, y, z;
} Point;

typedef struct polygon {
  int npoints;
  Point *points;
} Polygon;

typedef struct object {
  int npolygons;
  Polygon *polygons;
} Object;

typedef struct scene {
  int nobjects;
  Object *objects;
} Scene;
```

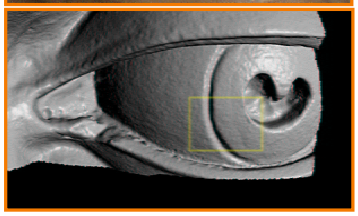


## Polygonal Meshes

- Can make complex objects

## Polygonal Meshes

- Can make very complex objects

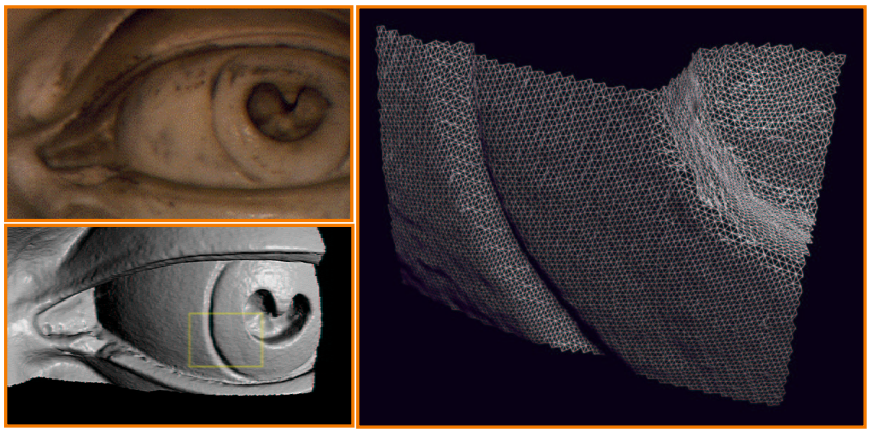


Stanford Graphics Laboratory

## Polygonal Meshes



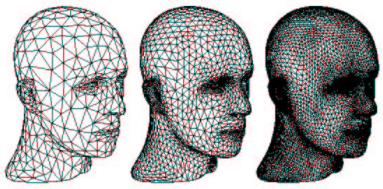
- Can make very complex objects



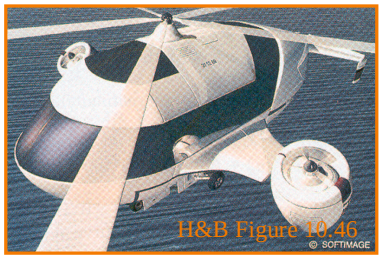
Stanford Graphics Laboratory

## More Geometric Representations





Subdivision Surfaces




Spline Surfaces

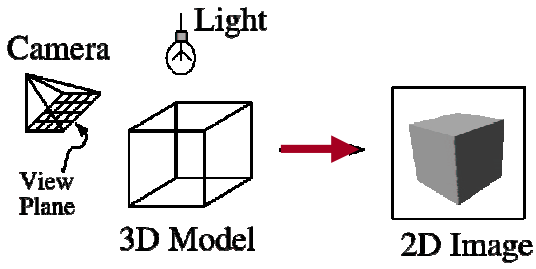
Stanford Graphics Laboratory

Solids

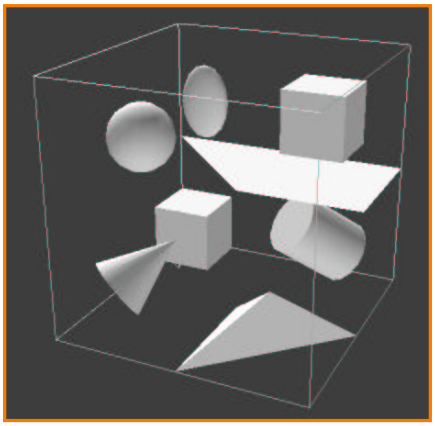
## Tin Toy


## Overview

- Topics in computer graphics
  - Imaging = *representing 2D images*
  - Modeling = *representing 3D objects*
  - Rendering = *constructing 2D images from 3D models*
  - Animation = *simulating changes over time*



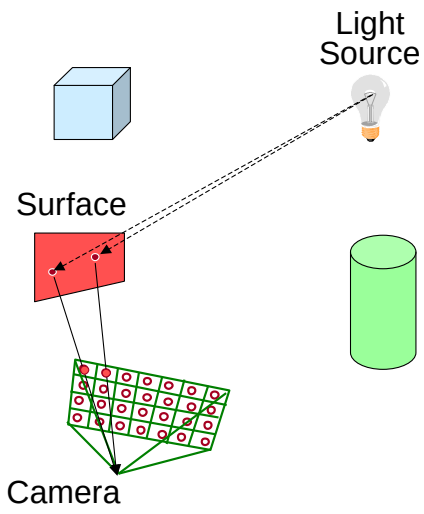
## Rendering Example



What issues must be addressed by a computer graphics rendering algorithm?

## Rendering Methods

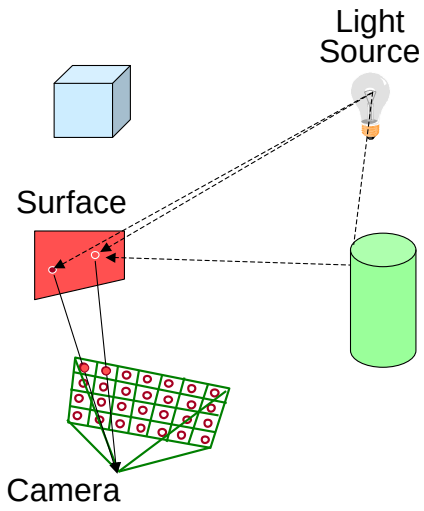
- Mathematical models for:
  - Light sources
  - Surface reflectances
  - Camera response
- Algorithms to find:
  - Visible surfaces
  - Indirect light paths
  - etc.



## Rendering Methods



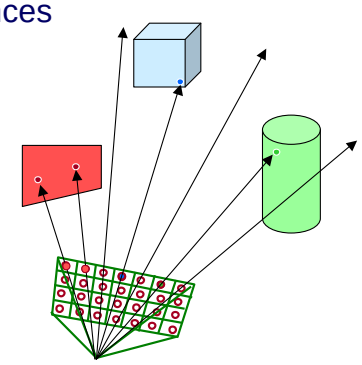
- Mathematical models for:
  - Light sources
  - Surface reflectances
  - Camera response
- Algorithms to find:
  - Visible surfaces
  - Indirect light paths
  - etc.




## Simple Rendering Algorithm

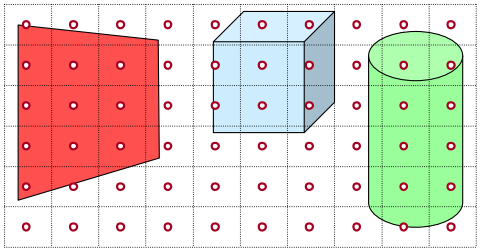


- For each pixel …
  - Construct ray from eye position through pixel
  - Find front-most surface intersected by ray
  - Compute color of sample based on models of light sources and surface reflectances



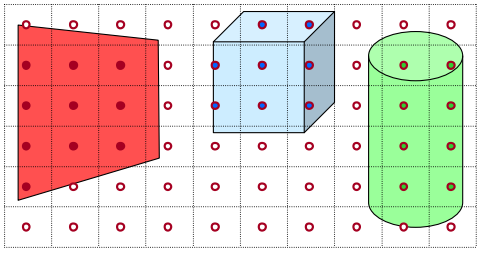
## Simple Rendering Algorithm

- For each pixel …
  - Construct ray from eye position through pixel
  - Find front-most surface intersected by ray
  - Compute color of sample based on models of light sources and surface reflectances



## Simple Rendering Algorithm

- For each pxiel …
  - Construct ray from eye position through pixel
  - Find front-most surface intersected by ray
  - Compute color of sample based on models of light sources and surface reflectances



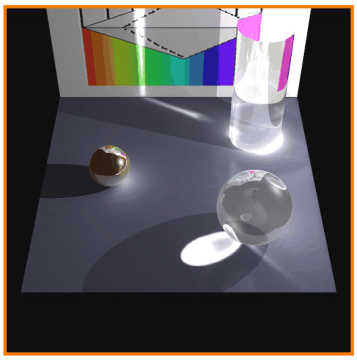
## Simple Rendering Algorithm

```
Image *RayCast(Camera *camera, Lights *lights, Scene *scene,
                    int width, int height)
{
    Image *image = Image_new(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Point intersection = FindIntersection(ray, scene);
            Color = ComputeReflectance(lights, intersection, scene);
            SetPixel(image, i, j, );
        }
    }
    return image;
}
```

## Better Rendering Algorithms

- Consider indirect light paths
  - Inter-object reflections
  - Caustics
  - etc.

Trike
*(James Percy, CS 426, Fall99)*

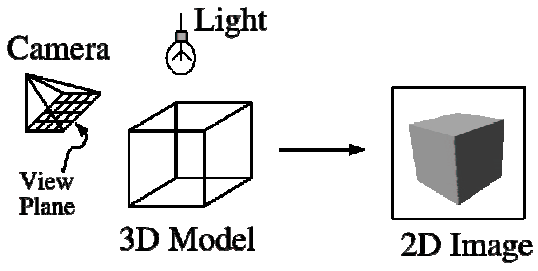


Rendering Caustics
*(Michael Bostock, James Percy & Casey McTaggert, CS 426, Fall99)*

## Red's Dream

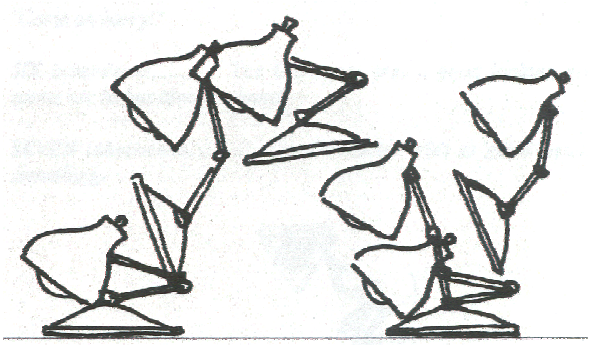

## Overview

- Topics in computer graphics
  - Imaging = *representing 2D images*
  - Modeling = *representing 3D objects*
  - Rendering = *constructing 2D images from 3D models*
  - Animation = *simulating changes over time*



## Animation

- Describing how models move
  - Kinematics
  - Dynamics
  - Planning
  - Learning

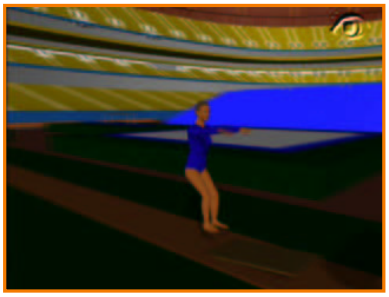


*(Lasseter87)*

## Animation



- Difficult combination of:
  - Desired behavior
  - Simulating physics
  - Artistic control



*Hodgins et al.*

*Chen, Guan, Liu, and Qie CS426, Fall98*

*Baraff & Witkin*

## Monsters, Inc.


## Computer Graphics Applications

- Entertainment
- Computer-aided design
- Scientific visualization
- Training
- Education
- E-commerce
- Computer art

## Computer Graphics Applications

- **Entertainment**
- Computer-aided design
- Scientific visualization
- Training
- Education
- E-commerce
- Computer art

Jurasic Park
*(Industrial, Light, & Magic)*

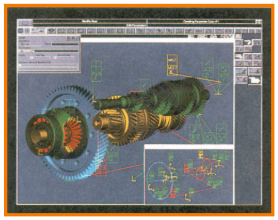
Geri's Game
*(Pixar Animation Studios)*

Quake
*(Id Software)*

## Computer Graphics Applications

- Entertainment
- **Computer-aided design**
- Scientific visualization
- Training
- Education
- E-commerce
- Computer art

Los Angeles Airport
*(Bill Jepson, UCLA)*

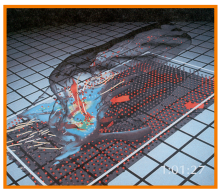


Gear Shaft Design
*(Intergraph Corporation)*

Boeing 777 Airplane
*(Boeing Corporation)*

## Computer Graphics Applications

- Entertainment
- Computer-aided design
- **Scientific visualization**
- Training
- Education
- E-commerce
- Computer art



Airflow Inside a Thunderstorm
*(Bob Wilhelmson, University of Illinois at Urbana-Champaign)*

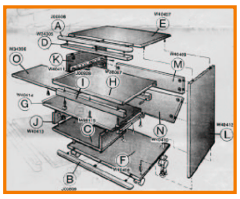
Apo A-1
*(Theoretical Biophysics Group, University of Illinois at Urbana-Champaign)*

Visible Human
*(National Library of Medicine)*

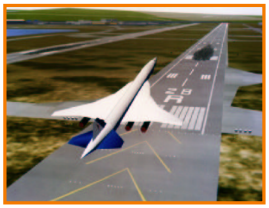
## Computer Graphics Applications

- Entertainment
- Computer-aided design
- Scientific visualization
- **Training**
- Education
- E-commerce
- Computer art

Driving Simulation
*(Evans & Sutherland)*



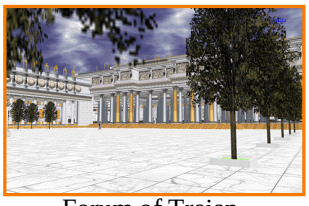
Desk Assembly
*(Silicon Graphics, Inc.)*



Flight Simulation
*(NASA)*

## Computer Graphics Applications

- Entertainment
- Computer-aided design
- Scientific visualization
- Training
- **Education**
- E-commerce
- Computer art



Forum of Trajan
*(Bill Jepson, UCLA)*



Human Skeleton
*(SGI)*

## Computer Graphics Applications

- Entertainment
- Computer-aided design
- Scientific visualization
- Training
- Education
- **E-commerce**
- Computer art

Interactive Kitchen Planner
*(Matsushita)*


Virtual Phone Store
*(Lucent Technologies)*

## Computer Graphics Applications

- Entertainment
- Computer-aided design
- Scientific visualization
- Training
- Education
- E-commerce
- **Computer art**

Blair Arch

*(Marissa Range & Adam Finkelstein, Princeton University)c*

## How Is This Related to the Course?

- Computer graphics uses …
  - Parsers (lexical and syntactic analysis)
  - Abstract data types
  - Memory management
  - Multiple processes
  - Networking

# How Is This Related to the Course?

- Computer graphics uses …
  - Parsers (lexical and syntactic analysis)
  - Abstract data types
  - Memory management
  - Multiple processes
  - Networking
  - Assembly code (maybe, a little)

Alone in the Dark 4
*(Darkworks/Infogrames)*